Virtual induction loops using smartphones for urban traffic control systems

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Transportation Research Procedia 00 (2016) 000–000

Transportation Research Procedia

www.elsevier.com/locate/procedia

International Conference on Intelligent Transport Systems in Theory and Practice, mobil.TUM 2017, 4-5 July 2017, Munich, Germany

Virtual induction loops using smartphones for urban traffic control systems

Guido Albertengo [a], Benedetta Cacioppo [a], Waqar Hassan [a,*], Eftychios Papapanagiotou [b]

[a] *Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Turin, Italy*
[b] *Chair of Traffic Engineering and Control, Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany*

Abstract

Growing traffic in ever congested urban cities can be managed by traffic control systems which monitor and forecast the traffic status and optimises the control strategy according to traffic flow. Such systems use input from inductive loop detectors which are costly to install and maintain. In this paper, we introduce and evaluate a novice software based solution called Virtual Induction Loops using a smartphone application and a central server. We also identify how current urban traffic control systems can benefit from such a dynamic solution.

Peer-review under responsibility of the scientific committee of the International Conference on Intelligent Transport Systems in Theory and Practice.

Keywords: virtual induction loops; urban traffic control; induction loops

1. Introduction

The latest advances in information technology open new possibilities for urban mobility. The digitalization of mobility (e.g. smartphone applications) does not only generate new services for the users but also generates great amounts of data stemming from the users. Particularly, urban intersections, being the hotspots of urban traffic networks, provide a very interesting case-study to demonstrate the potential of connected mobility. However, Urban Traffic Control (UTC) systems heavily rely on data gathered from induction loops (Anderson (1970)) to estimate the traffic situation and optimise the traffic flow and are not designed to include data from mobile devices. Leduc (2008) mentions that the installation, maintenance, and operation of these infrastructure-based detectors lead to substantial costs for municipalities and road operators. Moreover, in the case of faulty or inoperative induction loops, the performance of UTC system suffers significantly. In many cities, the percentage of induction loops that are out of service forces the operators to change the traffic signal control to fixed-time control, which leads to reduced performance.
This paper aims to supersede the induction loop detectors with Virtual Induction Loops (VIL), a feasible and practical software based solution which consists of an Android application and a central server in the cloud. While Gramaglia

* Corresponding author. Tel.: +39-011-090-4108.
E-mail address: waqar.hassan@polito.it

Peer-review under responsibility of the scientific committee of the International Conference on Intelligent Transport Systems in Theory and Practice.

et al. (2013) discusses a VIL solution based on cooperative vehicular communication, it requires placement of road-side units (RSU) as well as vehicles with GPS and capable of one-hop wireless communication. The goal of the paper is to study how a mobile application can be used as Virtual Induction Loop and how such an application can be functionally integrated with a UTC system and in particular in UTOPIA (Urban Traffic Optimization and Integrated Automation) described by Mauro and Di Taranto (1990).
The rest of this paper is organised as follows. Section 2 defines the concept used by Virtual Induction Loops and explains in detail the different techniques for the vehicle passage detection. Section 3 compares the results for the techniques used regarding the timestamp error, the execution time and success rate. Section 4 focuses on UTOPIA and how the concept of Virtual Induction Loops can be utilised and implemented. Finally, the paper concludes with the most important findings and future work.

2. Virtual Induction Loops

VIL is a system consisting of an Android application and a central server in the cloud. The application works in the background and is fully autonomous. Virtual loops are defined in the server at intersections of interest and are synchronised with the application. The application automatically detects when the user is on the move and intelligently samples location to identify transits over virtual loops and sends relevant information to the VIL server. The server aggregates and processes data from multiple users and forwards information such as vehicle count, passage time and passage speed to a UTC system.

2.1. Automatic trip start/stop detection

The Android application is able to detect that the user in on the move using one of the following techniques:

- Activity Recognition: using smartphone's sensors to detect the possible activity being performed by the user; refer to Park and Lee (2013)
- Bluetooth: based on connection/disconnection to/from a known Bluetooth device (such as vehicle's hands-free);
- Bluetooth beacon: based on the presence of a Bluetooth beacon emitted by a known Bluetooth beacon associated with a vehicle.

The application automatically samples location with a variable sampling period (to save energy) based on information from the above-mentioned techniques. For example, the application samples location with high sampling period (around 10 to 20 seconds) to save energy. On the other hand, when the application detects the device is in the vicinity of a virtual loop, it samples location with a higher frequency (every 2 to 4 seconds) to increase accuracy.

2.2. Definition of a virtual induction loop

In order to detect the passage of a vehicle through a goal or a virtual induction loop in real-time, which may coincide with a real induction loop, goals need to be defined. A database of all goals is defined using the following parameters.

- Goal ID: a unique identifier of the goal
- Latitude: in signed decimal degrees
- Longitude: in signed decimal degrees
- Bearing: in degrees

2.3. Goal passage timestamp evaluation

To identify the passage of a vehicle over a goal, we assess the distance between the vehicle and the goal and the bearing of the goal and vehicle. When a car is moving towards a goal, the distance between them will decrease. On the other hand, if the car is moving away, the distance will increase. At some point there will be two successive minimum

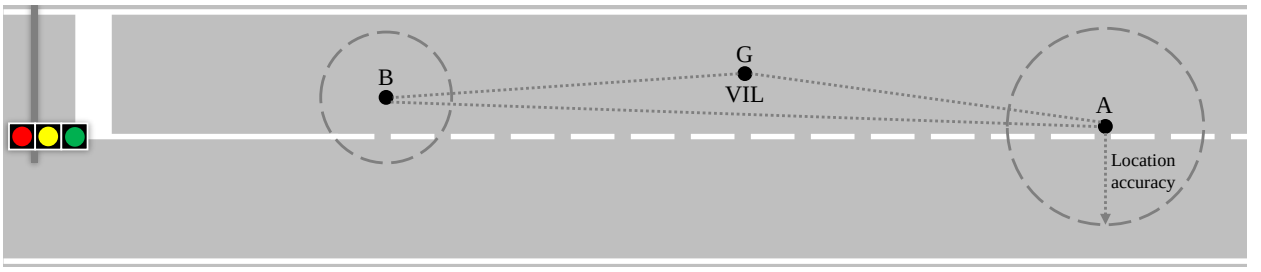


Fig. 1. A generic scenario showing GPS accuracy problem with two location points A and B and a goal point G

distances, one is computed just before the goal and the other one right after it. However, detecting these minimum distances is not sufficient, because the distance does not take into account the direction.
Each measurement should have a speed value and a timestamp value, v_0, t_0 before the goal and v_1, t_1 after the goal. Assuming a constant linear acceleration, the equation holds $a = \frac{\Delta v}{\Delta t} = \frac{v_1 - v_0}{t_1 - t_0}$. The acceleration can be used to utilise kinematic equations, which can be used to find the time t (the time of passage over point G). Based on the position of the car relative to the goal, there may be two different application modes *i.e.* forward mode and backward mode. In the forward mode, the vehicle is approaching the goal and moving forward towards it. In the backward mode, the vehicle is moving away from the goal. By carefully choosing the signs of the parameters, all possible cases can be addressed to find the unknown time of passage t.

2.4. GPS accuracy problem

Naturally, the accuracy of the time of passage evaluation heavily depends on the accuracy of the location information from GPS. Since the GPS is not always an exact system, it may return biased coordinates. A generic scenario of such a problem can be seen in Fig. 1. A and B are two location points with different accuracies and an offset error, whereas point G is the virtual induction loop under consideration. Coordinates of point G are known and are snapped to the road so that there is no error in its coordinates. Due to the error in location A and B, we need to compute the distance between the goal and the projected location of the vehicle on the stretch of road. To solve and evaluate this problem, we apply a number of different approaches.

2.4.1. First approach: Trigonometry

The first approach exploits trigonometric calculations. Let us analyse the triangle in Fig. 2a, the triangle formed by three points: the goal G, the last location before goal A and the first location after goal B. It is assumed that the bearing of the point A and B is the same. The distance needed to apply the forward mode is the result of a projection of the segment $\overline{AG}$ over the straight line passing through the bearing vector. In the same way, it is possible to compute the projection of the segment $\overline{GB}$ over the same straight line, needed to apply the backward mode.

Using the area of $\triangle ABG$ and law of sines, the angles α,β and γ can be found. The angle φ is computed as the difference between the bearing of G and B. The angle α_1 is computed by subtracting φ from α. The angle γ_1 is computed by subtracting α_1 and 90° from 180°. In the triangle $\triangle AKG$, the adjusted distance, namely the value of the segment $\overline{AK}$, is found applying the proportion given by the law of sines. Similarly, in backward mode the adjusted distance $\overline{BK}$ shown in Fig. 2b can be found.

2.4.2. Second approach: Plane geometry

The second approach exploits plane geometry. The x-axis is the equator line, the y-axis is the prime meridian (Greenwich) and a point on earth is determined by its latitude and longitude. Again the scenario consists of three points forming a triangular shape: the goal G, the point before the goal A and the one after B as shown in Fig. 3. The distance needed to apply the forward mode is the result of a projection of the segment $\overline{AG}$ over the straight line parallel to the bearing vector. The distance between K and A is the projection of the segment $\overline{AG}$.

First, we find the slope of straight lines parallel to bearing vector u, $m = \tan(\theta)$, where θ is the angle measured anticlockwise from the x-axis. Using the slope-intercept form of a line parrellel to bearing vector and passing through

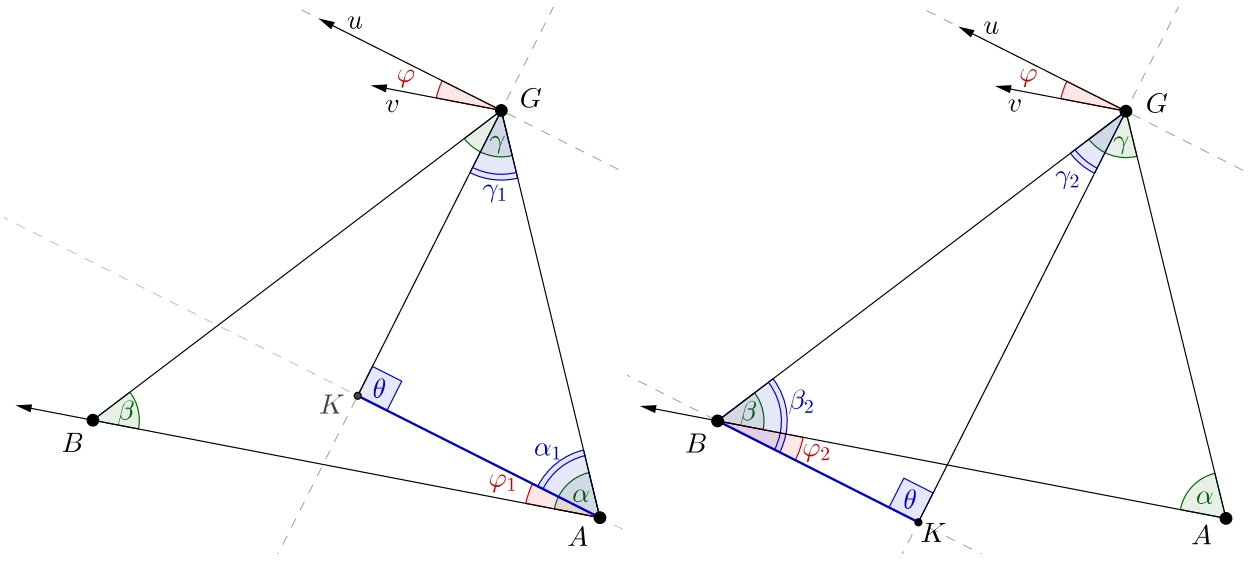


Fig. 2. Trigonometry application, portion (a) before and (b) after the goal

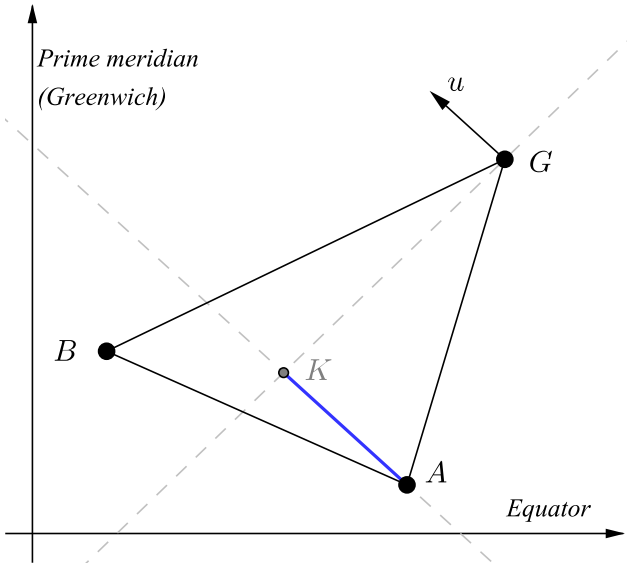


Fig. 3. Plane geometry application

A *i.e.* $y - y_A = m(x - x_A)$, and a line perpendicular to bearing vector and passing through G *i.e.* $y - y_G = -\frac{1}{m}(x - x_G)$, we can find the coordinates of K, y_K, x_K. Similarly, it is possible to compute the projection of the segment $\overline{GB}$, needed to apply the backward mode.

2.4.3. Third approach: Roto-translation

The third approach performs a rotation and translation centred around the goal as shown in Fig. 4a, where the black axes are the old reference system and the blue axes are the new reference system. All geographical coordinates are converted to Cartesian coordinates expressed in meters for getting more precise results.

First, the origin of the reference system is shifted to the goal, therefore the coordinates of the goal are subtracted

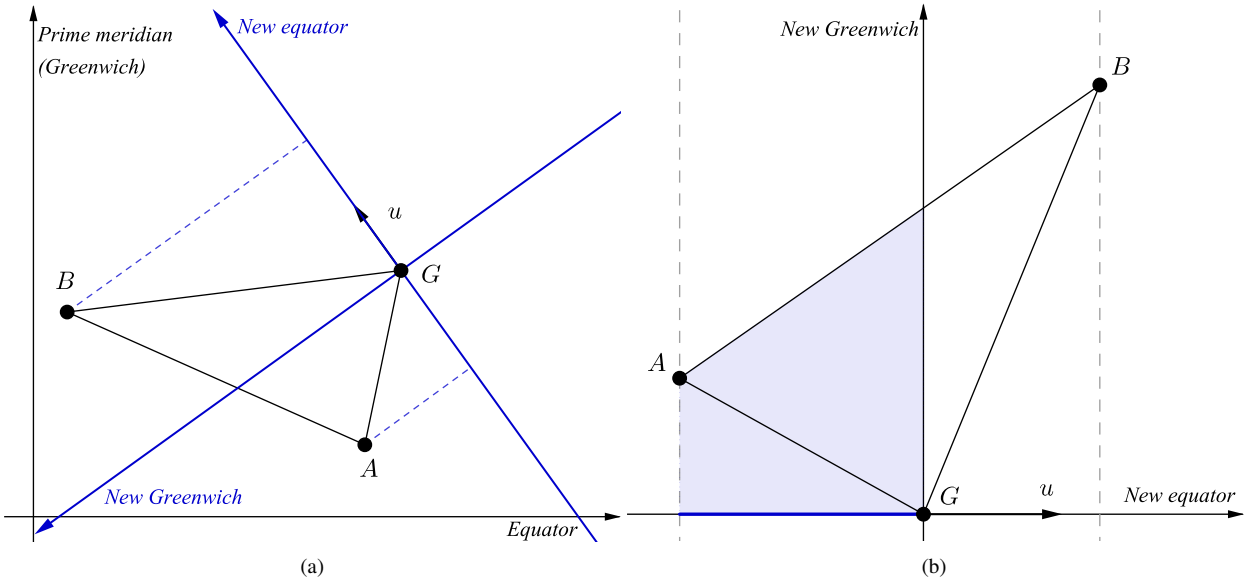


Fig. 4. (a) Pre and (b) Post roto-translation application

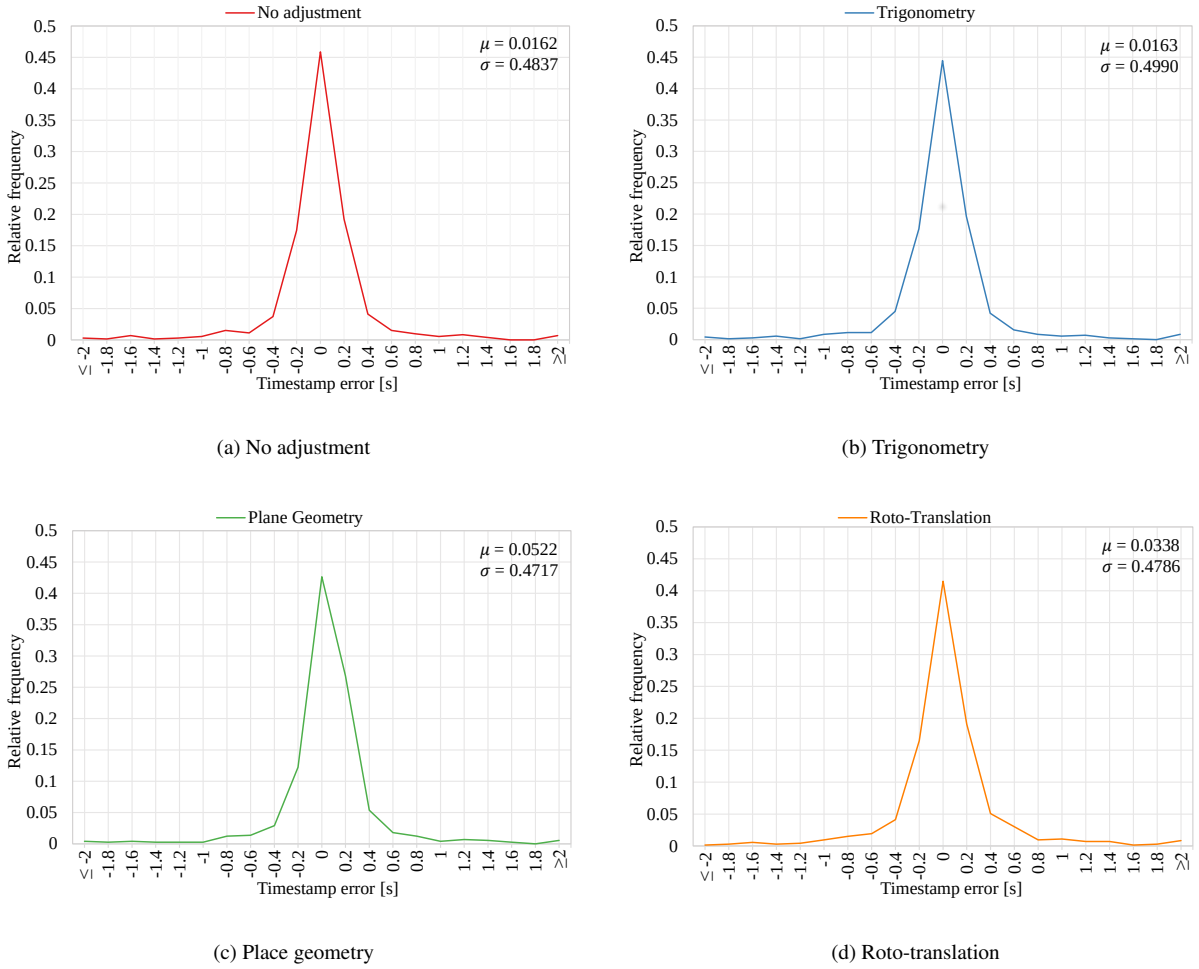
from the coordinates (x, y) of A and B using the translation equations $x' = x - x_G$ and $y' = y - y_G$. In this way the goal is at the origin and other points are located according to the new reference. Secondly, the reference system is rotated around G (the new origin) such that the rotation angle is $\theta = 90° - \beta$. The new rotated coordinates (X, Y) of the points A and B are computed as $X = x' \cdot \cos\theta + y' \cdot \sin\theta$ and $Y = -x' \cdot \sin\theta + y' \cdot \cos\theta$.
After the roto-translation the scenario appears like Fig. 4b. In the roto-translated reference system, the modulus of the x-component of A and B represents the distance between the point and the goal. The sign of the x-component indicates whether the point is before (negative) or after (positive) the goal.

3. Results

In order to prove the accuracy of the different approaches, a number of tests were performed to access the error in passage timestamp, CPU time and success rate. Location data collected from multiple users using an Android application was used as input for the tests. For the sake of conformity a filter was applied to the input data based on:

- Sample bearing must be within $\pm 10°$ of goal bearing. This ensures that only location points on a straight stretch of road are used and turns are avoided (since the bearing experiences significant change in a turn).
- Sample accuracy must be less than a threshold. Analysis on the test-set suggests a threshold of around 20-25m.
- Samples must not be placed too far apart *i.e.* $d(A, G) > r_A + r_G$.
- Samples must not be placed too near each other (such as nearly overlapped location points at a traffic signal).

Out of the test-set data, five random trips were selected for the tests. In every trip, after performing the above-mentioned filtering, tests were run. For every trip, every location sample is iterated over, considering three consecutive location samples at one time and taking the middle one was the goal. The computed timestamp is the average of the result from forward and backward mode, if both exist, otherwise the timestamp is equal to the only result successfully computed. Furthermore, a linear constant acceleration is assumed throughout the analysis.



(a) No adjustment

(b) Trigonometry

(c) Place geometry

(d) Roto-translation

Fig. 5. Error distribution in computed time of passage for different approaches

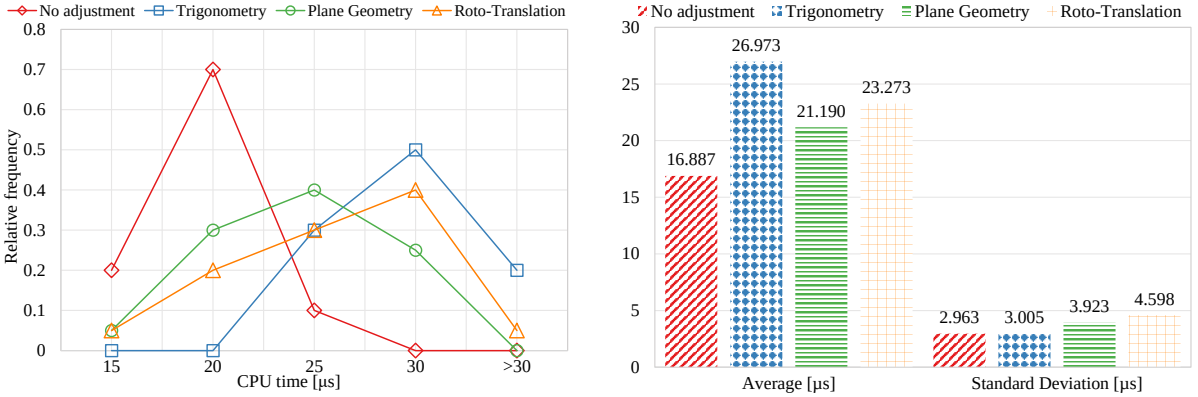
3.1. Timestamp error

Since three consecutive location samples (triplets) are used at once for analysis, the difference between computed timestamp (timestamp of passage) and actual timestamp can be easily calculated. A comparison is presented between the different approaches. Fig. 5 shows the distributions of the timestamp error using the four different methods over a random sample of more than 700 triplets. Each graph also shows the values of the average μ and standard deviation σ of the samples. The negative values in the distribution imply that the computed timestamp is smaller than the actual timestamp and vice versa for the positive ones. It is clear that the majority of the values are concentrated in the vicinity of zero. The values tend towards a bell-shaped curve and the curve is symmetric around zero. Comparing the four distributions there are no major differences. Only with plane geometry (Fig. 5c) there is a small trend of returning timestamps greater than the actual timestamp.

Table 1 lists the average (μ), minimum, maximum and standard deviation (σ) of timestamp errors achieved by all methods. The table shows that the average error is very small regardless of the method used, which is predictable since the distribution is symmetric around zero. The minimum error is quite low in each case, whereas apparently, the maximum error depends mainly on the nature of the trip since it is very similar for all methods. The standard deviations of the samples are not so high in all cases with respect to the average. This means that the values in all cases are quite close to the mean rather than spread out over a wide range.

Table 1. Average, minimum, maximum and standard deviations of timestamp error in seconds for each method

Method	Average (μ)	Min	Max	Standard deviation (σ)
No adjustment	0.0162	-4.62e-4	4.99	0.4837
Trigonometry	0.0163	-2.42e-6	5.00	0.4990
Plane Geometry	0.0522	1.72e-4	5.03	0.4717
Roto-translation	0.0338	0.0010	4.99	0.4786



(a) Distribution comparison

(b) Average and standard deviation comparison

Fig. 6. CPU execution time comparison for all algorithms

3.2. Execution time

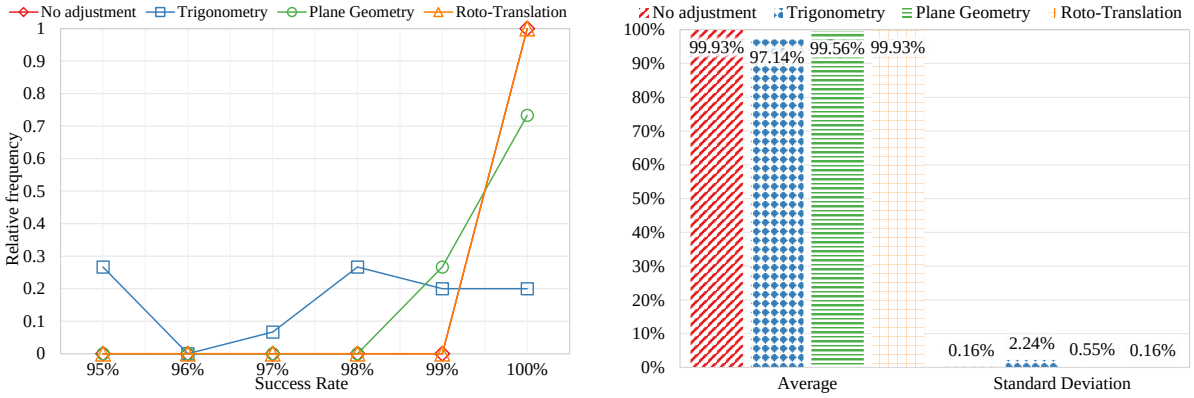
An important parameter to be analysed is the execution time for each method. The complexity of the algorithms influences the CPU execution time. To evaluate the execution time five random trips were selected as before and the algorithm was executed on all triplets in the data sample. All four variants of the algorithm were run one by one. To measure the CPU execution time of each method a Java Profiler was used, profiling each method separately in order to avoid inaccuracies due to the scheduling.

Fig. 6a shows the distribution of the CPU execution time of the four methods. The chart shows that the values are well distributed with a sharp peak concentrated around 20μs for the method with no adjustment. For other methods, the values have relatively flatter peaks concentrated between 25 to 30μs. Fig. 6b shows a comparison histogram of average and standard deviation of the CPU execution time for all four methods. The method with no adjustment is the fastest with the best average and low standard deviation, due to the fact that it does not involve any extra adjustment calculations. Methods with adjustments do not show diversity in averages and standard deviations. However, the analysis for other three methods is indecisive.

3.3. Success rate

The success rate is the fraction or percentage of success among a number of attempts. Due to the nature of mathematical calculations involved in the algorithms, it is possible that a method may result in a failure. As an example, when dealing with distances based on high accuracy latitude/longitudes, a fraction may have a numerator and/or denominator so small that it is approximated as a 0. This would result in a $\frac{0}{0}$, which is an undefined operation, returning NaN. Note that the returned timestamp is the average of the two timestamps from forward and backward modes if they both exist. Otherwise, it is equal to either of the valid timestamps. This makes success rate an important parameter to be analysed.

Fig. 7a shows the distribution of the success rate for all methods. The method with no adjustment and roto-translation



(a) Distribution comparison

(b) Average and standard deviation comparison

Fig. 7. Success rate comparison for all algorithms

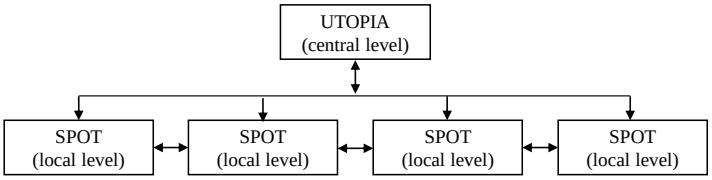


Fig. 8. Hierarchical architecture of UTOPIA/SPOT

exhibit similar behaviour with a concentration between 99% and 100%. Plane geometry method also shows concentrated distribution towards high success rate, whereas trigonometry method shows the worst performance, due to the nature of calculations. Fig. 7b shows the comparison of averages and standard deviations for all methods. The best success rate is achieved by the method without adjustment and roto-translation. Since no adjustment method and roto-translation show the same parameters, this implies that in this case, the few failures do not depend on the adjustment calculations, but rather on the kinematics mathematical model.

4. Integration

The information generated by the VIL system can he highly valuable to a UTC system. UTC systems consume data from conventional sources (such as induction loop detectors). In the following sections, we introduce a UTC system called UTOPIA and comment on the possible integration mechanism with UTOPIA.

4.1. UTOPIA/SPOT

UTOPIA (Urban Traffic Optimization and Integrated Automation) is an adaptive UTC system that has been successfully implemented in many cities around the world and optimises traffic flow by taking into account both private traffic and public transport vehicles. The goal of the optimisation is to minimise the total travel time in the network. UTOPIA is a closed loop control system, where the control actions (*i.e.* signal timings) are decided based on real-time traffic estimation. Induction loops are used to measure traffic flow and update the traffic estimation every 3 seconds, whereas traffic actuation takes place every second. To deal with the complexity of the signal optimisation problem in a traffic network consisting of numerous intersections, UTOPIA applies the closed loop approach in two levels (hierarchical system). The upper level (central level) monitors and controls the whole network consisting of areas with several intersections. The lower level (local level) controls single intersections while taking into account also adjacent intersections and the central level. The core intelligence of the system is located on the local level where the SPOT

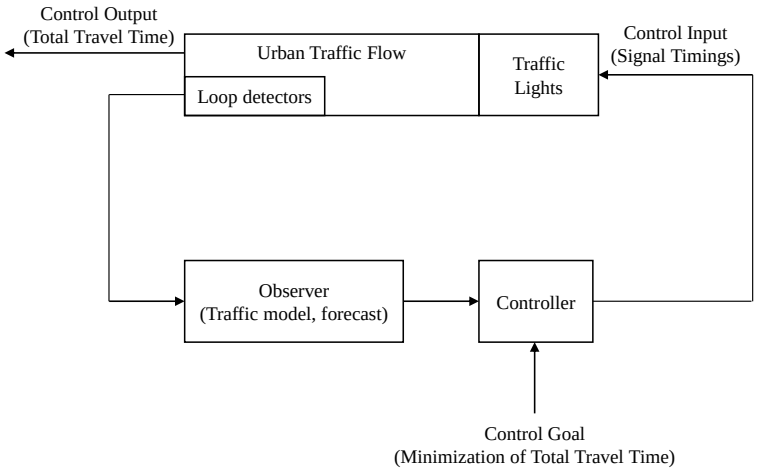


Fig. 9. Control loop of UTOPIA/SPOT

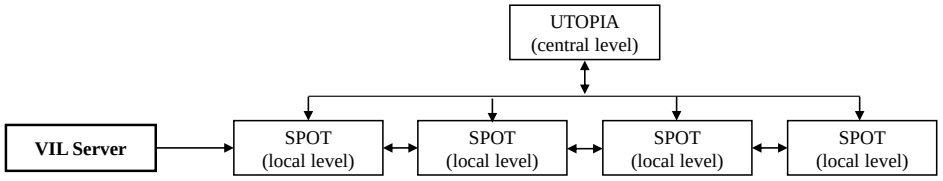


Fig. 10. Integration of VIL server at the local level

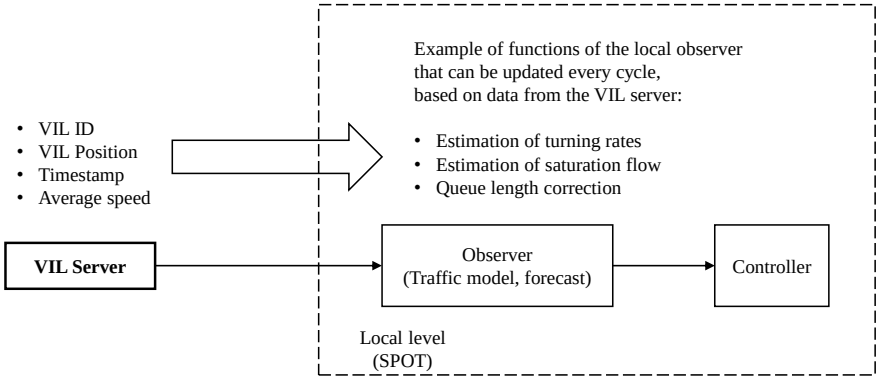


Fig. 11. Integration details of VIL at the local observer

unit plays the role of the local observer and controller. Fig. 8 shows the hierarchical approach of UTOPIA/SPOT and Fig. 9 depicts the control loop of UTOPIA/SPOT based on Mauro and Di Taranto (1990) and Papageorgiou et al. (2003).

4.2. Functional Integration of VIL

For the integration of Virtual Induction Loops in the UTOPIA/SPOT system, we propose in this paper the communication of the VIL server with the local units (SPOT) and not with the central level (UTOPIA) (refer to Fig. 10 and Fig. 11). The reason is that the local observer is already designed to update the estimation of certain crucial parameters every traffic signal cycle (typically around 70-90 seconds). If the VIL server contains information on an intersection that is proven to be more reliable than the information coming from stationary loop detectors it can be allowed to update the estimated values for certain cycles. This information will then be communicated also to the central level and

to the neighbouring intersections due to the inherent communication of the UTOPIA/SPOT system. The VIL should not be modelled as a typical induction loop in the SPOT unit, because of the completely different detection rate; we cannot expect that every passing vehicle is equipped with such an application. There are two possible approaches: Either the local observer has to be extended in order to "translate" the information from VIL into parameter estimation or the VIL server has to deliver the estimated parameter and its precision.

5. Conclusion

Intelligent traffic control systems are crucial for modern cities. The data source for these control systems has mainly been conventional induction loops which are expensive to install and maintain. While Gramaglia et al. (2013) discusses the possibility of switching traditional loops with virtual induction loops, the suggested VILs are also another piece of hardware. We propose a completely software based solution with already available and ubiquitous hardware, *i.e.* smartphones. The paper proves the feasibility of using smartphones to collect data and provide highly accurate information about a passage of a user over pre-defined goals. Simulation results based on real vehicular traces show negligible timestamp calculation errors with nearly 100% success rate. Moreover, due to flexible and scalable nature of VIL, the goals can be defined in real-time to focus on areas of special interest. We also proposed a practical integration scheme to allow UTC systems, such as UTOPIA, to benefit from VIL.

References

Anderson, R.L., 1970. Electromagnetic loop vehicle detectors. IEEE Transactions on Vehicular Technology 19, 23–30.

Gramaglia, M., Bernardos, C.J., Calderon, M., 2013. Virtual induction loops based on cooperative vehicular communications. Sensors 13, 1467–1476.

Leduc, G., 2008. Road traffic data: Collection methods and applications. Working Papers on Energy, Transport and Climate Change 1.

Mauro, V., Di Taranto, C., 1990. Utopia. In: Proceedings of the Sixth IFAC/IFIP/IFORS Symposium on Control and Communication in Transportation, Paris, France , 245–252.

Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., Wang, Y., 2003. Review of road traffic control strategies. Proceedings of the IEEE 91, 2043–2067.

Park, T., Lee, S., 2013. Sensor based activity detection. URL: `https://www.google.com/patents/US8560229`. US Patent 8,560,229.